Middlesex University London

Middlesex University Research Repository

An open access repository of

Middlesex University research

http://eprints.mdx.ac.uk

Busard, Simon, Pecheur, Charles, Qu, Hongyang and Raimondi, Franco ORCID logoORCID: https://orcid.org/0000-0002-9508-7713 (2019) Comparing approaches for model-checking strategies under imperfect information and fairness constraints. International Journal on Software Tools for Technology Transfer, 21 (4) . pp. 449-469. ISSN 1433-2779 [Article] (doi:10.1007/s10009-018-0505-6)

Final accepted version (with author's formatting)

This version is available at: https://eprints.mdx.ac.uk/25649/

Copyright:

Middlesex University Research Repository makes the University's research available electronically.

Copyright and moral rights to this work are retained by the author and/or other copyright owners unless otherwise stated. The work is supplied on the understanding that any use for commercial gain is strictly forbidden. A copy may be downloaded for personal, non-commercial, research or study without prior permission and without charge.

Works, including theses and research projects, may not be reproduced in any format or medium, or extensive quotations taken from them, or their content changed in any way, without first obtaining permission in writing from the copyright holder(s). They may not be sold or exploited commercially in any format or medium without the prior written permission of the copyright holder(s).

Full bibliographic details must be given when referring to, or quoting from full items including the author's name, the title of the work, publication details where relevant (place, publisher, date), pagination, and for theses or dissertations the awarding institution, the degree type awarded, and the date of the award.

If you believe that any material held in the repository infringes copyright law, please contact the Repository Team at Middlesex University via the following email address:

eprints@mdx.ac.uk

The item will be removed from the repository while any claim is being investigated.

See also repository copyright: re-use policy: http://eprints.mdx.ac.uk/policies.html#copy

Noname manuscript No.
(will be inserted by the editor)

Comparing Approaches for Model Checking Strategies under Imperfect Information and Fairness Constraints

Simon Busard · Charles Pecheur · Hongyang Qu · Franco Raimondi

the date of receipt and acceptance should be inserted later

Abstract Starting from *Alternating-time Temporal Logic*, many logics for reasoning about strategies in a system of agents have been proposed. Some of them consider the strategies that agents can play when they have partial information about the state of the system. $ATLK_{irF}$ is such a logic to reason about uniform strategies under unconditional fairness constraints. While this kind of logics has been extensively studied, practical approaches for solving their model-checking problem appeared only recently.

This paper considers three approaches for model checking strategies under partial observability of the agents, applied to $ATLK_{irF}$. These three approaches have been implemented in PyNuSMV, a Python library based on the state-of-the-art model checker NuSMV. Thanks to the experimental results obtained with this library and thanks to the comparison of the relative performance of the approaches, this paper provides indications and guidelines for the use of these verification techniques, showing that different approaches are needed in different situations.

Keywords alternating-time temporal logic · imperfect information · fairness constraints · model checking

Simon Busard
ICTEAM Institute, Université catholique de Louvain, Belgium
E-mail: simon.busard@uclouvain.be

Charles Pecheur
ICTEAM Institute, Université catholique de Louvain, Belgium
E-mail: charles.pecheur@uclouvain.be

Hongyang Qu
Department of Automatic Control and Systems Engineering, University of Sheffield, United Kingdom
E-mail: h.qu@sheffield.ac.uk

Franco Raimondi
Department of Computer Science, Middlesex University, United Kingdom
E-mail: f.raimondi@mdx.ac.uk

1 Introduction

Alternating-time Temporal Logic (ATL) is a logic for reasoning about the strategies of agents in multi-agent systems [1]. It assumes that all agents have full information about the current state of the system through its execution. Some extensions of this logic remove this assumption and reason about the strategies of agents that have only a partial view of the system (see for instance [24,21,16], and a comparison of strategic abilities in [4]). $ATLK_{irF}$ is a logic that reasons about the strategies of memoryless agents with imperfect information—we speak about *uniform strategies*—in a system with unconditional fairness constraints [8].

Let us consider a simple example inspired by the card game of [24]. The game is played between a *player* and a *dealer*. It is played with three cards, an *Ace*, a *King* and a *Queen*, such that the Ace wins over the King, the King wins over the Queen, and the Queen wins over the Ace. The game is played in two steps. First, the dealer gives one card to the player, takes one card for himself and puts the last one on the table, hidden from the player. Second, the player can choose to change his card with the one on the table, or to keep it. After this decision, the game stops, the cards are revealed and the winner is the one with the winning card.

In this game, the player has no uniform strategy to win. Whatever the player chooses to do, the dealer can give cards that make the player lose. Nevertheless, if we extend the game by introducing infinite repetition and we assume that the dealer will give the cards in a fair way by giving each pair infinitely often, the player has a strategy to win eventually. He knows that, since the dealer is fair, he will eventually receive a winning card. $ATLK_{irF}$ allows us to reason about the uniform strategies of the player under the assumption that the dealer is fair. It can also be used to reason about the strategies of multi-agent programs under the control of a fair scheduler [14].

A large body of research has been developed on the theoretical issues of logics reasoning about uniform strategies (see for example [23] for the complexity of checking the existence of such strategies, or [15] for the undecidability of the variant with perfect recall). Nevertheless, practical approaches for solving their model-checking problem appeared only recently [27,20,8].

This paper experimentally compares the three approaches of Pilecki et al. [27], Huang and van der Meyden [20], and Busard et al. [8] in a unified framework based on binary decision diagrams (BDDs) [3]. The three techniques have been adapted to the setting of $ATLK_{irF}$ and implemented in PyNuSMV [6], a Python framework based on the state-of-the-art model checker NuSMV [12]. This paper compares the efficiency of these implementations on three models taken from the literature and draws conclusions on the strengths and drawbacks of the approaches. The tests show that there is no best approach for all cases, and that each technique is better than the others in different situations. Based on these conclusions, a set of guidelines are proposed to choose the best approach when facing a new model-checking problem.

The remainder of the paper is organised as follows: Section 2 presents the logic $ATLK_{irF}$, its syntax, models and semantics, and briefly describes the three approaches and their implementation. Section 3 presents the models and properties on which the tests have been conducted, and Section 4 analyses the results. Finally, Section 5 concludes.

2 Background

This section presents an overview of the syntax and semantics of $ATLK_{irF}$. It then briefly describes the three approaches compared in this paper and discusses their implementation with PyNuSMV.

2.1 Reasoning about uniform strategies under fairness constraints

The logic $ATLK_{irF}$ combines atomic propositions, propositional logic operators ($\neg$, $\wedge$,...), branching-time temporal operators (EX, AF,...) [13], knowledge operators (K_{ag}, C_Γ,...) [18] and strategic operators ($\langle\!\langle\Gamma\rangle\!\rangle X$, $[\![\Gamma]\!]F$,...) [1].

More precisely, given a set of atomic propositions AP and a set of agents Ag, $ATLK_{irF}$ formulas follow the grammar

$$\begin{aligned}\phi &::= true \mid p \mid \neg\phi \mid \phi \vee \phi \mid E\psi \mid \langle\!\langle\Gamma\rangle\!\rangle\psi \mid \\ &\quad K_{ag}\phi \mid E_\Gamma\phi \mid D_\Gamma\phi \mid C_\Gamma\phi \\ \psi &::= X\phi \mid \phi\ U\ \phi \mid \phi\ W\ \phi\end{aligned}$$

where $p \in AP$, $ag \in Ag$ and $\Gamma \subseteq Ag$. The other common propositional, temporal and strategic operators can be derived from the previous ones:

$$\begin{aligned} false &\equiv \neg true \\ \phi_1 \wedge \phi_2 &\equiv \neg(\neg\phi_1 \vee \neg\phi_2) \\ \phi_1 \implies \phi_2 &\equiv \neg\phi_1 \vee \phi_2 \\ \phi_1 \iff \phi_2 &\equiv (\phi_1 \implies \phi_2) \wedge (\phi_2 \implies \phi_1) \\ A\psi &\equiv \neg E\neg\psi \\ [\![\Gamma]\!]\psi &\equiv \neg\langle\!\langle\Gamma\rangle\!\rangle\neg\psi \\ F\phi &\equiv true\ U\ \phi \\ G\phi &\equiv \phi\ W\ false \end{aligned}$$

Intuitively, $E\psi$ means that there exists a path starting at the current state satisfying ψ. On the other hand, $A\psi$ means that all paths starting at the current state satisfy ψ. For example, $EX\phi$ means that there exists a successor of the current state satisfying ϕ and $AF\phi$ means that all paths starting at the current state will eventually reach a state satisfying ϕ.

Furthermore, $K_{ag}\phi$ means that agent ag knows that ϕ is true in the current state, $E_\Gamma\phi$ means that every agent in Γ knows that ϕ is true, $D_\Gamma\phi$ means that, by sharing their knowledge of the current state, agents of Γ know that ϕ is true, and $C_\Gamma\phi$ means that ϕ is common knowledge among agents of Γ.

Finally, $\langle\!\langle\Gamma\rangle\!\rangle\psi$ means that the agents of Γ have a strategy to ensure that all paths resulting from playing this strategy will satisfy ψ. For example, $\langle\!\langle\Gamma\rangle\!\rangle G\phi$ means that the agents of Γ have a strategy to enforce ϕ forever. On the other hand, $[\![\Gamma]\!]\psi$ means that the agents of Γ do not have a strategy to achieve ψ.

2.1.1 Models and notations

$ATLK_{irF}$ formulas are interpreted over *imperfect information concurrent game structures with fairness constraints* (iCGSf in short). These structures extend standard (imperfect information) concurrent game structures with unconditional fairness constraints [1,29]. More precisely, given a set AP of atomic propositions, an iCGSf is a structure $S = \langle Ag, Q, Q_0, Act, e, \delta, V, \sim, FC\rangle$ such that

- Ag is a finite set of *agents*;
- Q is a finite set of *states*;
- $Q_0 \subseteq Q$ is the set of *initial* states;
- Act is a finite set of *actions*; a *joint action* is a tuple of actions of Act, one for each agent of Ag;
- $e : Ag \to (Q \to (2^{Act} \setminus \emptyset))$ defines, for each agent $ag \in Ag$ and state $q \in Q$, the set of actions ag can choose in q, that is, actions *enabled* in q; we write e_{ag} for the function $e(ag)$ returning the actions ag can choose in any state;
- $\delta : Q \times Act^{|Ag|} \nrightarrow Q$ is a partial deterministic *transition function* defined for each state $q \in Q$ and each joint action enabled in q; we write $q \xrightarrow{a} q'$ for $\delta(q,a) = q'$;

- $V : Q \to 2^{AP}$ is a function labelling states with atomic propositions from AP;
- $\sim: Ag \to 2^{Q \times Q}$ defines a set of *equivalence classes* representing the observability of agents; we assume that each agent chooses its actions based on its own knowledge of the system, that is,

$$\forall q,q' \in Q, q \sim_{ag} q' \implies e_{ag}(q) = e_{ag}(q') \tag{1}$$

  for any agent ag;
- $FC \subseteq 2^Q$ is a set of *fairness constraints*.

Given a set of agents $\Gamma \subset Ag$, $\sim^D_\Gamma = \bigcap_{i \in \Gamma} \sim_{ag}$ is the *distributed knowledge* relation; $\sim^E_\Gamma = \bigcup_{i \in \Gamma} \sim_{ag}$ is the *group knowledge* relation; $\sim^C_\Gamma$, defined as the reflexive transitive closure of the relation $\sim^E_\Gamma$, is the *common knowledge* relation.

We say that a joint action $a \in Act^{|Ag|}$ *completes* an action $a_\Gamma \in Act^{|\Gamma|}$ for a set of agents Γ, written $a_\Gamma \sqsubseteq a$, if the action for each agent of Γ in a is the same as the action of the same agent in a_Γ. Given a joint action $a \in Act^{|Ag|}$ and a set of agents $\Gamma \subseteq Ag$, we write $a(\Gamma)$ for the tuple of actions of agents of Γ in a. When $\Gamma = \{ag\}$ is a singleton, we write $a(ag)$ instead of $a(\{ag\})$. We call a Γ-*move* an element $\langle q, a_\Gamma \rangle \in Q \times Act^{|\Gamma|}$ such that $\forall ag \in \Gamma, a_\Gamma(ag) \in e_{ag}(q)$, that is, a pair composed of a state and a joint action for Γ enabled in the state.

A *path* is an infinite sequence $\pi = q_0 \xrightarrow{a_1} q_1 \xrightarrow{a_2} \ldots$ such that $\delta(q_d, a_{d+1}) = q_{d+1}$ for all $d \geq 0$. We write $\pi(d)$ for q_d. A path in S is *fair* if it meets all fairness constraints of S infinitely often, that is, π is fair if, for each fairness constraint $fc \in FC$, there exist infinitely many indices d such that $\pi(d) \in fc$. A state q is *reachable* in a structure S if there exists a path π in S such that $\pi(0) \in Q_0$ and there exists $d \geq 0$ such that $\pi(d) = q$. A state q is *fair* if there exists a fair path starting at q. A fair reachable state is thus a state belonging to a fair path starting at an initial state of S; such a state is considered, by each agent of the structure, as possibly reached through a valid run of the structure.

A *memoryless strategy* for a given agent ag is a function $f_{ag} : Q \to Act$ such that $\forall q \in Q, f_{ag}(q) \in e_{ag}(q)$. Intuitively, a strategy for agent ag tells the agent which action to choose in each state of the system. A *uniform* strategy for agent ag is a strategy f_{ag} such that for all $q,q' \in Q$, if $q \sim_{ag} q'$ then $f_{ag}(q) = f_{ag}(q')$. Intuitively, a uniform strategy is a strategy that needs only the current observations of the agent to be played. These strategies correspond to strategies that the agent can effectively play [24]. In the sequel, when speaking about strategies that are not necessarily uniform, we speak about *general* strategies.

We call *outcomes* of a strategy the set of paths of the structure that are coherent with the strategy. More precisely, the outcomes of a strategy f_{ag} for agent ag from a state q are defined as

$$out(f_{ag}, q) = \left\{ \begin{array}{l} \pi = q_0 \xrightarrow{a_1} q_1 \xrightarrow{a_2} \ldots \mid \\ q_0 = q \wedge \forall d \in \mathbb{N}, f_{ag}(q_d) \sqsubseteq a_{d+1} \end{array} \right\}. \tag{2}$$

Finally, a (uniform) strategy for a group of agents $\Gamma \subseteq Ag$ is a set of (uniform) strategies, one for each agent of Γ. The outcomes of a strategy f_Γ for a group of agents Γ from a state q are defined as

$$out(f_\Gamma, q) = \bigcap_{f_{ag} \in f_\Gamma} out(f_{ag}, q). \tag{3}$$

These outcomes are the paths that are coherent with every strategy of the set f_Γ. A strategy f_Γ can be represented as a set of Γ-moves as

$$\{\langle q, a_\Gamma \rangle \in Q \times Act^{|\Gamma|} \mid a_\Gamma = f_\Gamma(q)\}, \tag{4}$$

that is, the set of moves such that the actions for Γ are the ones specified by the strategy.

2.1.2 Semantics

The semantics of $ATLK_{irF}$ formulas is defined over states of a structure S by the relation $S, q \models \phi$; S is omitted when clear from the context. This relation is the standard semantics for propositional operators, branching-time operators and knowledge operators. In particular, an agent ag (or a group Γ) knows that ϕ is true in state q if ϕ is true in every possibly reached state q' indistinguishable from q by ag (or Γ). For strategic operators, the relation quantifies over uniform strategies. More precisely, the relation $q \models \phi$ is defined as

$$\begin{array}{ll}
q \models true & \\
q \models p & \Leftrightarrow p \in V(q) \\
q \models \neg\phi & \Leftrightarrow q \not\models \phi \\
q \models \phi_1 \vee \phi_2 & \Leftrightarrow q \models \phi_1 \text{ or } q \models \phi_2 \\
q \models E\psi & \Leftrightarrow \left\{ \begin{array}{l} \text{there exists a } \textit{fair path } \pi \text{ such that} \\ \pi(0) = q \text{ and } \pi \models \psi \end{array} \right. \\
q \models K_{ag}\phi & \Leftrightarrow \left\{ \begin{array}{l} q' \models \phi \text{ for all } q' \text{ s.t. } q \sim_{ag} q' \\ \text{and } q' \text{ is a } \textit{fair reachable} \text{ state} \end{array} \right. \\
q \models \mathscr{K}_\Gamma\phi & \Leftrightarrow \left\{ \begin{array}{l} q' \models \phi \text{ for all } q' \text{ s.t. } q \sim^{\mathscr{K}}_\Gamma q' \\ \text{and } q' \text{ is a } \textit{fair reachable} \text{ state,} \\ \text{where } \mathscr{K} \in \{D,E,C\} \end{array} \right. \\
q \models \langle\!\langle \Gamma \rangle\!\rangle \psi & \Leftrightarrow \left\{ \begin{array}{l} \text{there exists a } \textit{uniform strategy } f_\Gamma \text{ s.t.} \\ \text{for all } ag \in \Gamma, \text{ for all } q' \sim_{ag} q, \\ \text{for all } \textit{fair paths } \pi \in out(f_\Gamma, q'), \\ \pi \models \psi. \end{array} \right.
\end{array}$$

The relation $\pi \models \psi$ over paths π of the structure S is defined as

$$\pi \models X\phi \quad \Leftrightarrow \pi(1) \models \phi$$

$$\pi \models \phi_1 U \phi_2 \quad \Leftrightarrow \begin{cases} \text{there exists } d \geq 0 \text{ s.t. } \pi(d) \models \phi_2 \\ \text{and for all } e < d, \pi(e) \models \phi_1 \end{cases}$$

$$\pi \models \phi_1 W \phi_2 \quad \Leftrightarrow \begin{cases} \text{there exists } d \geq 0 \text{ s.t. } \pi(d) \models \phi_2 \\ \text{and for all } e < d, \pi(e) \models \phi_1, \\ \text{or for all } d \geq 0, \pi(d) \models \phi_1. \end{cases}$$

While the logic integrates temporal and epistemic operators to allow users to express a large range of properties, the remainder of this paper focuses on strategic operators only.

2.2 Model-checking techniques

This section briefly presents the three approaches for model checking $ATLK_{irF}$ strategic formulas compared in this paper. It also describes the idea of pre-filtering surely losing moves that can make them faster. This section is intentionally short and informal as formally describing these three approaches is not a contribution of this paper. The interested reader can find such formal descriptions in [5].

2.2.1 The partial approach

The first model-checking approach considered in this paper has been proposed by Busard et al. [8]. In the sequel, it is called the *partial* approach. It is based on the notion of partial uniform strategies, that is, uniform strategies that are not defined for all states of the structure. Given a set of states of interest for which we want to find winning uniform strategies—such as the initial states—, the partial approach generates all partial strategies with a forward traversal of the structure. More precisely, this forward traversal alternates between discovering new states in which we need to make a (uniform) choice, and splitting the possible choices into non-conflicting ones. Each generated partial strategy is then evaluated using custom fixpoint computations to determine whether it is winning for the states of interest or not. The whole approach is valid because partial strategies generated from the states of interest are sufficient to determine whether there exists a uniform strategy that is winning in these states.

The partial approach has been improved by using two simple practical optimisations. The first one is caching: for each partial strategy, the approach needs to evaluate sub-formulas of the considered strategic formula to determine whether it is winning or not. But these sub-formulas can contain strategic formulas themselves, that are re-evaluated again and again in states belonging to several partial strategies. To avoid recomputing the truth value of these sub-formulas again and again, the approach caches the results. The second optimisation is early termination: as the approach is interested in getting the states of interest for which there exists a winning uniform strategy, it can stop as soon as one has been found for each state. These two simple optimisations have been shown to be very useful in practice [8].

2.2.2 The early approach

The second approach is based on the one proposed by Pilecki et al. [27]. Their main idea is that we do not need to extend a partial strategy in all states that matter before concluding whether it is winning or not in the states of interest. More precisely, we can check if all extensions of the current partial strategy are winning, and this is easily performed using custom fixpoint computations. If all extensions of the current partial strategy are winning, we know that there exists a winning uniform strategy and can stop there.

The approach used in this paper—called the *early* approach in the sequel—extends the original idea of Pilecki et al. in several ways. First, in addition to checking whether all extensions of the current partial strategy are winning, it also tries to filter out states for which we know no winning extension exists. More precisely, we can check whether there exists a winning general strategy that extends the current partial strategy, with other custom fixpoint computations. If there is none, we can stop extending the current strategy and explore other choices.

Second, the approach of Pilecki et al. is limited to one initial state and one top-level strategic operator. These limitations introduce simplifications in the problem they solve: they can stop the process as soon as they find a winning strategy for this single initial state, instead of keeping track of the states for which they already know the truth value. Furthermore, considering only one state prevents the technique to be applied to sub-formulas, for which we need to consider the same strategy in different states. The early approach handles any subset of states of interest, making it usable to check strategic sub-formulas in subsets of states.

Third, the original approach of Pilecki et al. does not handle fairness constraints, while the early one does. Finally, caching and early termination can also be applied to the early approach.

2.2.3 The symbolic approach

The last approach is based on the one proposed by Huang and van der Meyden [20]. It is called the *symbolic* approach in the sequel. Their main idea is to derive, from the structure under consideration, a new structure where the strategies are encoded in the derived states themselves. Then, fixpoint computations are performed on the derived structure to compute all the strategies that are winning for the state they are embedded into. These fixpoint computations are easily performed, but the derived structure is exponentially larger than

the original one as it embeds all possible uniform strategies in its states.

The symbolic approach of this paper is a particular case of the approach of Huang and van der Meyden [20]. Their work describes a logic that explicitly quantifies over strategies of agents [19]. Thanks to their logic, it is, for instance, possible to express the fact that an agent has a strategy to ensure two objectives at the same time.

On the other hand, the symbolic approach is tailored to the verification of $ATLK_{irF}$. More precisely, the original approach of Huang and van der Meyden is tailored by restricting the syntax of the logic to $ATLK_{irF}$. Furthermore, the original work of Huang and van der Meyden defines so-called *strategic agents* to be able to reason about states of the derived structure that share the same strategy. In our setting, these strategic agents are not necessary. They are thus omitted.

Finally, in the context of the symbolic approach, caching and early termination make no sense because sub-formulas are evaluated only once and all winning strategies are computed at the same time.

2.2.4 Pre-filtering surely losing moves

The three approaches above can be improved by pre-filtering losing moves. More precisely, it is easy to compute the moves that belong to a winning *general* strategy—with some custom fixpoint computations—, and if some move does not belong to a winning general strategy, then it does not belong to a winning *uniform* one. Thus we can remove, from the structure, all the moves that do not belong to a general winning strategy before applying any of the model-checking techniques. The approach works with the three considered approaches: removing surely losing moves leads to fewer partial strategies to consider, and to fewer uniform strategies to encode in the derived structure, reducing its size.

2.3 Implementation

The approaches have been implemented with PyNuSMV [6], a Python framework giving access to internal functionalities of NuSMV, a state-of-the-art symbolic model checker [12], such as model construction and BDD manipulation. It facilitates the implementation of new BDD-based model-checking algorithms, such as the ones considered in this paper. This section briefly describes the language used to describe the models and the actual implementation of the approaches.

2.3.1 Modeling language

The modeling language supported by the implementation is based on the NuSMV language. The model of an iCGSf is composed of a standard NuSMV model and a description of its agents directly in Python. The NuSMV model describes the states and actions of the structure, its transition function, its labelling function and its fairness constraints. It does not define the agents of the iCGSf, nor what they can observe and how they can act. Instead, these agents are defined as Python instances, with their name, the set of variables of the NuSMV model they observe and the set of input variables representing their actions. These agents also define the actions that are enabled for them in each state—through the transition function of the NuSMV model—and the equivalence classes.

Additional conditions must be met by the NuSMV model to correctly represent an iCGSf:

- the sets of actions of agents must be disjoint, that is, they must control different actions of the model;
- for each agent and each state of the system, the enabled actions are not constrained by the actions of another agent;
- for each agent and each equivalence class—defined by the variables it observes—the enabled actions are the same in all the states of the equivalence class.

If these additional conditions are met by the NuSMV model, it correctly represents an iCGSf. All the models presented in this paper meet these conditions.

For instance, Figure 1 shows the NuSMV model corresponding to the card game of the Introduction. It is composed of the `Player` and `Dealer` modules defining the actions enabled in each state for both agents, and the `main` module instantiating both modules, defining the top-level variables such as the cards of the players `pcard` and `dcard` (the `odcard` variable is used for the player to observe the card of the dealer only in the final step), and the current `step` of the game. It also defines the initial states and the transition relation by describing how state variables evolve according to the actions of the agents. It finally declares six fairness constraints to model a fair dealer that will give all pairs of cards infinitely often. More information on the NuSMV modeling language can be found in the user manual [11].

Figure 2 presents the declaration of the two agents in Python. The first agent of the model, called `player`, observes the `step`, `pcard`, and `odcard` state variables, and controls the `player.action` input variable. The second agent is called `dealer`, observes all the state variables, and controls the `dealer.to_player` and `dealer.to_dealer` input variables.

2.3.2 Implementation of the approaches

The approaches are well suited to work within a BDD-based framework. In particular, iCGSf are naturally encoded with BDDs [20]. Furthermore, the *partial* and *early* approaches perform an enumeration of strategies, so we can encode each strategy as a BDD. The fixpoint computations the two approaches use can be implemented as operations on the BDDs

```
MODULE Player(step)
    IVAR action : {none, keep, swap};
    TRANS action in
          (step = 1 ? {keep, swap} : {none})

MODULE Dealer(step)
    IVAR to_player : {none, Ac, K, Q};
         to_dealer : {none, Ac, K, Q};
    TRANS step = 0 -> (to_player != to_dealer &
                       to_player != none &
                       to_dealer != none)
    TRANS step != 0 -> (to_player = none &
                        to_dealer = none)

MODULE main
    VAR step   : 0..2;
        pcard  : {none, Ac, K, Q};
        dcard  : {none, Ac, K, Q};
        odcard : {none, Ac, K, Q};
        dealer : Dealer(step);
        player : Player(step);

    DEFINE
    win := step = 2 & ( (pcard = Ac & dcard = K) |
                        (pcard = K & dcard = Q)  |
                        (pcard = Q & dcard = Ac) );

    INIT step = 0 & pcard = none &
         dcard = none & odcard = none

    TRANS next(step) = (step + 1) mod 3

    TRANS step = 0 -> next(pcard) = dealer.to_player
    TRANS step = 1 -> case player.action = keep :
                               next(pcard) = pcard;
                          TRUE :
                               next(pcard) != pcard &
                               next(pcard) != dcard &
                               next(pcard) != none;
                      esac
    TRANS step = 2 -> next(pcard) = none

    TRANS step = 0 -> next(dcard) = dealer.to_dealer
    TRANS step = 1 -> next(dcard) = dcard
    TRANS step = 2 -> next(dcard) = none

    TRANS step != 1 -> next(odcard) = none
    TRANS step = 1 -> next(odcard) = dcard


    FAIRNESS step = 1 & pcard = Ac & dcard = K
    FAIRNESS step = 1 & pcard = Ac & dcard = Q
    FAIRNESS step = 1 & pcard = K  & dcard = Ac
    FAIRNESS step = 1 & pcard = K  & dcard = Q
    FAIRNESS step = 1 & pcard = Q  & dcard = Ac
    FAIRNESS step = 1 & pcard = Q  & dcard = K
```

Fig. 1 The NuSMV model of the iCGSf of the card game.

representing different parts of the model (the initial states, the transition function, etc.) and the strategies.

The *symbolic* approach can be implemented in a fully symbolic way, in the sense that, given an iCGSf, we can build the derived structure and encode it with BDDs in the same way we encode any iCGSf. More precisely, the uniform strategies of the agents are encoded with new variables in the model. That is, for each agent *ag* and set of states indistinguishable for *ag*, a new state variable is declared, taking its values in the set of actions agent *ag* can play in this equivalence class. A uniform strategy for *ag* is thus represented as the choice *ag* makes in each equivalence class of the model. As an optimisation, only the strategies of agents appearing in coalitions of the checked formula are encoded, as the strategies of the others have no relevance in this case.

```
agents = {Agent("player",
                {"step", "pcard", "odcard"},
                {"player.action"}),
          Agent("dealer",
                {"step", "pcard", "dcard", "odcard"},
                {"dealer.to_player",
                 "dealer.to_dealer"})}
```

Fig. 2 The Python-defined agents of the iCGSf of the card game.

When the derived structure is encoded, the different operations composing the approach can be implemented as operations on the BDDs representing the different parts of the derived model, without needing to explicitly represent the different strategies. This advantage of reducing the model-checking problem to fixpoint computations and manipulations of BDDs is at the cost of encoding strategies as variables in the derived structure. The resulting structure is then much larger than the original one, making the different operations on BDDs more costly.

These implementations are are available at `http://lvl.info.ucl.ac.be/Tools/PyNuSMV`.

3 Models and properties

To compare the relative performances of the different approaches, we used three models with strategic formulas. This section describes the three models, how they are encoded as iCGSf, and the formulas we verified.

$ATLK_{irF}$ includes temporal and epistemic operators in addition to strategic ones. However, these temporal and epistemic operators have a negligible impact on the performances of algorithms checking strategic operators: temporal and epistemic sub-formulas can be evaluated in negligible time using standard techniques and be provided to strategic-operator-related algorithms as they were atomic propositions. The formulas considered in this paper thus focus on strategic operators only.

3.1 Tian Ji and the king

The first model is based on the ancient Chinese tale of Tian Ji [25]. The model contains two agents, the king and his

general Tian Ji, playing a horse racing game. Each agent has N horses $h_1, ..., h_N$ such that if the king plays with horse h_i against Tian Ji with horse h_j, the winner is the one with the highest index. If $i = j$, then the winner is chosen non-deterministically. The game is composed of N races and one horse can play only once. The winner of the game is the player with the most won races. The game is replayed infinitely.

The formal iCGSf of the game includes two agents, *king* and *Tian Ji*. The states of the model are composed of the remaining horses for Tian Ji and for the king, and the score of both players. In the initial states, all horses are available for both players and both scores are null. There is one action per horse, and each player can play any action corresponding to the remaining horses. Furthermore, each player observes only his remaining horses, not the other's, that is, the equivalence classes for a player are such that two states are indistinguishable if the states share the same remaining horses for the player. Finally, fairness constraints are specified such that the king chooses his horses in any order, infinitely many times each. This is achieved by asking the king to choose the order of his horses before starting the game, and specifying one fairness constraint per such an initial order. The number of reachable states grows exponentially in terms of the number of horses.

The number of strategies of Tian Ji, depending on the number of horses N, is given by the equation:

$$\prod_{i=1..N} i^{C_N^i},$$

where C_N^i is the number of combinations of i elements among N. Indeed, in each state where Tian Ji has i remaining horses, he can choose one of them. Furthermore, there are C_N^i different sets of i remaining horses that Tian Ji can have, giving the equation above. For example, Tian Ji has 24 strategies with 3 horses and 20736 ones with 4 horses. The symbolic approach has to consider all these strategies, while the others can reduce their number by restricting the search for partial strategies and by pre-filtering out losing moves.

Figure 3 shows a simplified version of the game of Tian Ji, where the king always chooses his weakest horse for the next race and the scores are not tracked. In this simplified game, Tian Ji has 24 complete uniform strategies. Indeed, he can choose three different horses in the initial state, and two different horses in states of the second row. In the states of the third and fourth rows, he has to choose the last remaining horse, or do nothing to restart the game. This amounts to $3*2*2*2*1*1*1*1 = 24$ uniform strategies. On the other hand, he has 6 partial strategies from the initial state as, when he chose to run with his strongest horse during the first race, for instance, the choices he would make in states in which he did not make this particular choice have no impact on whether the strategy is winning or not. Figure 3 illustrates such a partial strategy in bold. Tian Ji has the same number of uniform strategies in the standard game as he does not observe the choices of the king nor the tracked scores.

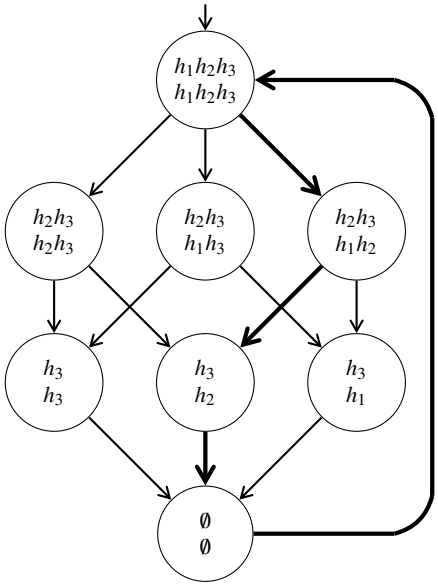

Fig. 3 A simplified version of the Tian Ji game. States are labelled with the remaining horses for the king (first line) and for Tian Ji (second line). Actions of the agents are easily inferred.

The first formula checked on this model is

$$\phi_1^T = \langle\langle \textit{Tian Ji} \rangle\rangle \mathbf{F}\ \textit{Tian Ji wins},$$

where *Tian Ji wins* is true in all states where there are no remaining horses and Tian Ji has a higher score than the king. This formula says that Tian Ji can eventually win a game. It is true in the initial states of the model. Indeed, since the king is fair and will choose any possible order infinitely often, Tian Ji can play the same combination over and over again, and this combination will be winning at some point. In fact, any memoryless uniform strategy for Tian Ji is winning.

Two other formulas have been checked on this model to highlight the differences of behaviours of the tested approaches. The second formula is

$$\phi_2^T = \langle\langle \textit{Tian Ji} \rangle\rangle \mathbf{F}\ \langle\langle \textit{Tian Ji} \rangle\rangle [\neg \textit{King wins}\ \mathbf{U}\ \textit{Tian Ji wins}].$$

Similarly to *Tian Ji wins*, *King wins* is true in all states where there are no remaining horses and the king has a higher score than Tian Ji. This formula says that Tian Ji has a strategy to reach states in which he can surely win the current game, that is, a state in which he can win before the king wins. The formula is not true in the model because there exists no state in which Tian Ji is sure to win the current game. He does not observe the current score, so even in states in which he wins the current game, he does not see it. Furthermore, he does not see the current order of the king's horses, and cannot adapt his choices accordingly.

The last formula checked on the model of Tian Ji is

$$\phi_3^T = \langle\langle \mathit{Tian\ Ji} \rangle\rangle \mathbf{X}\ \mathit{Tian\ Ji\ null\ score},$$

where *Tian Ji null score* is true in all states where the score of Tian Ji is 0. This formula is not true in the initial states of the model because Tian Ji has no horse that he is sure will lose the race. Even the slowest horse h_1 could possibly win the race if the king also plays with h_1.

3.2 The three castles

The second model is derived from [27]. It is composed of three castles with corresponding health points ranging from 0 to 3, 0 health points meaning that the castle is defeated. Each castle is defended by a set of workers. At each turn, a worker can attack another castle, defend his own castle or do nothing, but a worker cannot defend his castle twice in a row. The number of damages a castle receives is the number of attackers against this castle minus the number of defenders of this castle, if it is greater than 0. The health points of the castles are not reset at each turn, so the game is played in several turns. Finally, the workers only observe whether they can defend their castle or not, and, for each castle, whether it is defeated or not. The model is parametrised with the number of workers of each castle.

The formal model contains one agent per worker. The states are composed of the health points of each castle and whether or not each worker can defend his castle at the next turn. In the initial state, all castles have 3 health points and all workers can defend their castle. The actions of each worker are (1) doing nothing—the only one enabled when his castle is defeated—(2) defending his castle—only possible when he did not defend it in the previous turn—(3) for each other castle, one action to attack it. The equivalence classes are such that for a given worker, he can only observe whether he can defend his castle and, for each castle, whether it is defeated or not. Finally, a unique fairness constraint is defined, containing all states of the model. This ensures that all paths of the model are fair. The initial state is specially marked such that all workers can distinguish it from any other state. This is necessary to be able to specify that some workers have a strategy in the initial state for a given objective. Without this, they would not be able to differentiate the initial state from other states in which the castles have fewer health points. As for Tian Ji's model, the number of reachable states of the model grows exponentially in the number of workers.

Each worker has 82944 possible strategies. These strategies can be decomposed as his choices in

- the initial state, in which he can choose one action among four: doing nothing, defending his castle or attacking one of the two other castles. This amounts to 4 possible choices.
- the non-initial states in which his castle is not defeated and he did not defend it in the previous turn. There are four equivalence classes matching this case, depending on whether the other castles are defeated or not. In these cases, the worker can choose between the four actions. This amounts to $4^4 = 256$ possible combinations of choices.
- the non-initial states in which his castle is not defeated but he defended it in the previous turn. There are four equivalence classes matching this case. The worker can choose between three actions as he cannot defend his castle. This amounts to $3^4 = 81$ possible combinations of choices.
- the non-initial states in which his castle is defeated. There are 8 equivalence classes matching this case, depending on whether the other castles are defeated and whether he defended his castle just before or not. The worker can choose only one action, giving $1^8 = 1$ possible choice.

All these choices can be combined in any way since they are exclusive in the equivalence classes they consider, giving $4 * 256 * 81 * 1 = 82944$ possible uniform strategies for one worker.

The depth of the model—that is, the number of steps needed to reach all the reachable states from the initial one—does not change with the number of workers since it depends only on the health points of the castles. An exception is when there are few workers, that is, with one worker in each castle. In this case, the depth is a bit higher because there are too few workers to ensure to quickly reach a final state. The partial and early approaches really depend on this depth since it dictates how far the partial strategies are.

We are interested in two formulas. The first one is

$$\phi_1^C = \langle\langle \mathit{Castle_1}, \mathit{Castle_2} \rangle\rangle \mathbf{F}\ \mathit{Castle_3\ defeated},$$

where Castle_i groups the workers of the ith castle and the atomic proposition *Castle₃ defeated* is true in all states in which the third castle has 0 health points. This formula is true in all tested models, but is not true in general. If the third castle has enough workers, they are able to defend the castle and prevent the other workers to damage it. More precisely, if the third castle has more workers than the addition of the two others, the formula is false, even if the workers have perfect information. The tested models always have enough workers in the first two castles to make the formula satisfied.

The second formula is

$$\phi_2^C = \langle\langle \mathit{Worker_1}, \mathit{Worker_2} \rangle\rangle \mathbf{F}\ \mathit{all\ defeated},$$

where *Worker₁* (resp. *Worker₂*) is a worker of the first castle (resp. second castle), and *all defeated* is true in the states where all castles have 0 health points. This formula is false in all tested models because, even if they can defeat the third castle, the workers have not enough information to ensure that the two other castles will be defeated at the same time.

3.3 The prisoners and the light bulb

The third model is based on the problem of the 100 prisoners and the light bulb [17]:

> "A group of 100 prisoners, all together in the prison dining area, are told that they will be all put in isolation cells and then will be interrogated one by one in a room containing a light with an on/off switch. The prisoners may communicate with one another by toggling the light switch (and that is the only way in which they can communicate). The light is initially switched off. There is no fixed order of interrogation, or interval between interrogations, and the same prisoner may be interrogated again at any stage. When interrogated, a prisoner can either do nothing or toggle the light switch, or announce that all the prisoners have been interrogated. If that announcement is true, the prisoners will (all) be set free, but if it is false, they will be executed. While still in the dining room, and before the prisoners go to their isolation cells (forever), can the prisoners agree on a protocol that will set them free?"

One way to guarantee their freedom is to designate a *counter* among the prisoners. This counting prisoner starts at 0 and each time he enters the room and the light bulb is switched on, he switches it off and increments his counter. Every time another prisoner enters the room, if the light bulb is switched off and he has never switched it on, he switches it on before leaving the room. When the prisoner with the counter enters the room and his counter is at 99, he is sure that all prisoners have entered the room at least once and he can safely announce that all prisoners have visited the room. This strategy is winning if the warden fairly chooses the prisoners each day because the counter will enter infinitely often, thus will be able to switch the light off as many times as he wants, and the other prisoners will also enter the room infinitely often and be able to switch the light on once.

The formal model encodes this problem of the prisoners and the light bulb, designates a special prisoner that can keep track of a counter and gives the ability to the other prisoners to remember whether they already switched the light on or not. The idea behind the model is to verify that the prisoners effectively have a strategy to be released with these limited capabilities.

More precisely, the formal model is composed of the warden and N prisoners; one of them is the counting one. The warden keeps track of which prisoners have been interrogated at least once—to be able to release or execute them when the counting prisoner makes an announce—and chooses the next prisoner to interrogate. Furthermore, he keeps track of whether the prisoners should be released (if the counting prisoner correctly announces that all prisoners have been interrogated) or executed (if the counting prisoner makes an incorrect announce). The counting prisoner keeps track of his counter and each prisoner keeps track of whether he has already switched the light bulb or not. The states of the model are thus composed of the state of the light bulb (on or off), who has already been interrogated, should the prisoners be released or not, and executed or not, the counter of the counting prisoner, whether each prisoner has already switched the light bulb or not, and finally the prisoner that is being currently interrogated. In the initial state, the light bulb is off, nobody has been interrogated, the prisoners should not be released or executed, the counter is at 0, no prisoner has already switched the light bulb, and nobody is currently interrogated. When nobody is interrogated, the warden can choose any prisoner, and when a prisoner is interrogated, he can choose to switch the light bulb or do nothing, and the counting prisoner can additionally choose to make the announcement, as well as to increment his counter. Each prisoner sees the light bulb when he is currently interrogated, knows whether he already switched the light or not, and knows whether he is currently interrogated. In addition, the counting prisoner knows the value of his counter. All prisoners (including the counting one) can distinguish the initial state from the others. Finally, fairness constraints are specified such that each prisoner is interrogated infinitely often. Again, the number of reachable states of the model grows exponentially in terms of the number of prisoners.

In this model, the counting prisoner has 6^{2N} uniform strategies, where N is the number of prisoners (including himself). Indeed, he can do nothing when he is not interrogated, so only his choices when he is interrogated can lead to different strategies. The model contains $2N$ equivalence classes for the counting prisoner when he is currently interrogated. Indeed, he observes the value of his counter (from 0 to $N-1$) and the state of the light bulb. In each class, he can perform $3*2=6$ different actions: doing nothing, switching the light bulb, or making an announcement, and furthermore incrementing his counter or not. This gives us 6^{2N} possible strategies for the counting prisoner: 1296 strategies with two prisoners, 46656 ones with three prisoners. The other prisoners have 16 different strategies. Such a prisoner can only do something when he is interrogated. In this case, the model contains 4 equivalence classes as he observes the state of the light bulb and whether or not he already switched it on. In each of these classes, the prisoner can switch the bulb or not, giving us 2^4 possible strategies. So there are at most $16^{N-1}*6^{2N}$ strategies to consider when checking whether the coalition of the N prisoners (including the counting one) can enforce a given objective.

We are interested in the formula

$$\phi^P = \langle\langle prisoners \rangle\rangle [\neg executed \, \mathbf{U} \, released], \tag{5}$$

saying that the prisoners have a collective strategy to be released before being executed. This formula is true in the

model, showing that the prisoners effectively have a counting strategy to be free.

4 Measures and comparisons

The formulas of the previous section have been checked using the approaches on models of increasing size. This section presents and compares the results. All the experiments have been performed on a MacBook Pro with a 2.6GHz processor and 16GB RAM, and under a time limit of 1800 seconds. This time limit is indicated by a horizontal line in the graphs, and data points reaching this time limit are depicted above the line. Each data point is the average of 20 runs. The observed variability was very low for all measurements. Furthermore, these experiments usually consumed less than 1GB of memory, but some consumed up to several GBs. They nevertheless never consumed all the available memory.

In all the figures, short names are used to refer to the tested approaches:

- *Partial* refers to the partial approach with caching and early termination, but without pre-filtering;
- *Partial/filt* refers to the partial approach with caching, early termination, and pre-filtering;
- *Early* refers to the early approach with caching and early termination, but without pre-filtering;
- *Early/filt* refers to the early approach with caching, early termination, and pre-filtering;
- *Symbolic* refers to the symbolic approach without pre-filtering;
- *Symbolic/filt* refers to the symbolic approach with pre-filtering.

The partial and early approaches use caching and early termination. For the former, [8] showed that using both always increases performances, and early experiments not provided in this paper showed the same results for the latter.

For each approach, observations are given, then the differences of performances are explained based on these observations.

The Python implementation used for the following experiments is a prototype showing the applicability of the approaches. It would not compete with dedicated tools performing the same kind of tasks. These experiments are not meant to show the absolute performances of the implementation but the relative gain of the different approaches.

4.1 Tian Ji and the king

This section presents the results for the properties of Tian Ji's model.

*4.1.1 $\langle\langle$Tian Ji$\rangle\rangle$**F** Tian Ji wins*

Figure 4 shows the evolution of the verification time in terms of the number of horses, for the six approaches checking the formula ϕ_1^T on the model of Tian Ji. As explained before, any strategy of Tian Ji is winning for this objective because the fair king will ensure that all configurations will eventually happen.

Pre-filtering Pre-filtering removes no move because any general strategy is winning.

Partial approaches The Partial approach checks only one strategy. Since all of them are winning, early termination allows the approach to stop its processing after one strategy.

The Partial/filt approach also checks one strategy. Pre-filtering removes no move, so the extra effort is not beneficial. Nevertheless, the time needed for pre-filtering is not negligible. But dynamic reordering of BDD variables accelerates the process of checking the single strategy, thanks to the better variable order computed during pre-filtering.

Early approaches The Early approach extends the first strategy to $\lfloor \frac{N}{2} \rfloor + 1$ steps before concluding it is a winning strategy. At smaller steps, the approach cannot decide whether all extensions are winning because Tian Ji did not win enough races to be sure to win the game.

The Early/filt approach also extends the first strategy to $\lfloor \frac{N}{2} \rfloor + 1$ steps before concluding it is a winning strategy, as pre-filtering is not beneficial. Pre-filtering is not negligible, but the dynamic reordering of variables accelerates the rest of the process.

Symbolic approaches The Symbolic approach encodes and tests all strategies at once. The Symbolic/filt approach behaves exactly like Symbolic since pre-filtering removes no move. In this case, the time needed for pre-filtering is negligible.

Comparison First, pre-filtering removes no moves, thus is useless in reducing the number of strategies to consider. The time needed to perform pre-filtering in the Symbolic/filt approach is negligible compared to the time needed for checking the strategies. This explains why both versions of this approach are the same.

On the other hand, pre-filtering takes a significant amount of time in the partial and early cases, as the number of checked strategies is very small. But the time needed by both versions (with and without pre-filtering) are very similar. This is explained, for both approaches, by the fact that pre-filtering, while removing no moves, triggers the dynamic reordering of the BDD variables. The new order computed after pre-filtering is substantially better than the initial one

Fig. 4 Evolution of the verification time of the approaches for the formula $\phi_1^T = \langle\!\langle \textit{Tian Ji} \rangle\!\rangle \mathbf{F}\ \textit{Tian Ji wins}$.

and allows the verification of strategies to be performed faster. This can be shown by disabling dynamic reordering. In this case, the approaches with pre-filtering take more time to check the formula, and the difference is the time needed to perform the useless pre-filtering. The fact that the approaches with and without pre-filtering perform similarly is thus purely coincidental.

Second, the Partial approach is better than the Early approach because Partial checks only one partial strategy while Early needs to check $\lfloor \frac{N}{2} \rfloor + 1$ sub-models before concluding that there is a winning strategy.

The symbolic approaches have worse performances than the others. This is explained by the fact that, as all strategies are winning and the other approaches stop as soon as a winning strategy is found, symbolic approaches have to perform more work to check all strategies at the same time.

4.1.2 $\langle\!\langle$Tian Ji$\rangle\!\rangle$ F $\langle\!\langle$Tian Ji$\rangle\!\rangle$[¬King wins U Tian Ji wins]

Figure 5 shows the evolution of verification time for the six approaches on the formula ϕ_2^T checked on the model of Tian Ji. As said before, this formula is not satisfied by the initial states of the model because the inner strategic formula $\langle\!\langle \textit{Tian Ji} \rangle\!\rangle [\neg \textit{King wins}\ \mathbf{U}\ \textit{Tian Ji wins}]$ is false in all states. On the other hand, Tian Ji has a winning strategy if he can observe the whole system because he knows which order the king will play for the current game.

Pre-filtering Empirically, we can observe that pre-filtering removes about 50% of the moves for the strategic sub-formula. Tian Ji has a winning general strategy in many states, but not all actions lead to winning the game. Furthermore, pre-filtering really helps for the top-level strategic formula because no state satisfies the sub-formula, making the top-level one trivially unsatisfiable, even with perfect information.

Partial The Partial approach checks all partial strategies for the sub-formula

$$\langle\!\langle \textit{Tian Ji} \rangle\!\rangle [\neg \textit{King wins}\ \mathbf{U}\ \textit{Tian Ji wins}]$$

in all reachable states as the partial strategies for the initial states lead to all reachable states. Furthermore, since the top formula is false in the initial states, the approach checks all partial strategies for the initial states for the top formula. Thus, it has to check all partial strategies, for both strategic sub-formulas, to conclude that the formula is false.

Nevertheless, this approach has to check the sub-formula for relatively small subsets of the reachable states each time, since partial strategies reach relatively small subsets of the reachable states. Indeed, as the top-level strategies do not cover the complete set of reachable states—for instance, if Tian Ji chooses his best horse, he will not consider the states in which he still can play this horse—, the number of strategies to consider for the sub-formula (for this particular top-level strategy) is not large.

Partial/filt Pre-filtering in the Partial/filt approach triggers the evaluation of the sub-formula in all the reachable states at once. This represents a large number of partial strategies, compared to computing strategies for different separate subsets of states as for the Partial approach. More precisely, as all reachable states are considered at the same time, all choices remain in all equivalence classes, thus the number of strategies is not reduced.

Nevertheless, when the evaluation of sub-formula is done, the approach detects that there are no general winning strategies in the initial states, thus it can directly conclude that the formula is violated.

Early The Early approach evaluates N strategies for the top formula because there are N possible initial moves for Tian Ji. It needs to check this small amount of strategies (for the

Fig. 5 Evolution of the verification time of the approaches for the formula $\phi_2^T = \langle\langle \textit{Tian Ji} \rangle\rangle \mathbf{F}\ \langle\langle \textit{Tian Ji} \rangle\rangle [\neg \textit{King wins}\ \mathbf{U}\ \textit{Tian Ji wins}]$.

top formula) because, for each of them, it evaluates the sub-formula in the reached states and immediately determines that there is no extending winning strategy. On the other hand, it checks the sub-formula on a large subset of the reachable states because the top-formula-related strategies reach a large number of states.

The sub-formula is evaluated on many states at once. This means that a large partial strategy is already reached when splitting the moves in these states. This large partial strategy is sufficient to determine that the formula is false in these states. Thus, the Early approach has to check several strategies, but these strategies are determined as losing without extending them.

Early/filt The Early/filt approach triggers pre-filtering for the top formula, that evaluates the sub-formula on all the reachable states. This means that the approach has to check the sub-formula for all these states at once. Evaluating the sub-formula on all the reachable states also triggers pre-filtering for the sub-formula removing about half the moves.

Furthermore, the number of strategies (splitting the other half of the moves) in the whole set of reachable states is large. Nevertheless, the Early/filt approach can directly determine that each strategy is losing, avoiding to extend them completely.

The number of strategies to check in the whole set of reachable states is way larger than the strategies checked by the Early approach. Nevertheless, when the sub-formula has been evaluated on all the reachable states, pre-filtering for the top formula determines that no state is winning and no strategy must be checked for the top formula.

Symbolic The Symbolic approach checks all the strategies at once. First, it computes the set of states satisfying the sub-formula, then the set of states satisfying the top formula. It encodes strategies for Tian Ji once, even if there are two strategic formulas, because it can reuse the encoded strategies for both formulas.

Symbolic/filt The Symbolic/filt approach first triggers pre-filtering. This leads to checking the sub-formula. This sub-formula is first pre-filtered, but nothing is gained because any action that is ruled out in a state can be winning in another indistinguishable state. Nevertheless, the approach does not have to encode the strategies for the top formula because pre-filtering evaluates that there are no possibly winning moves, directly determining that the top formula cannot be true.

Comparison Pre-filtering in the Partial/filt approach evaluates the sub-formula for all reachable states at once, representing a large number of strategies. On the other hand, the Partial approach evaluates the sub-formula on smaller subsets of states. These states reach only a subset of the states of the model, reducing the number of strategies to consider. Overall, this lazy evaluation allows the Partial approach to compute fewer strategies than Partial/filt to evaluate the sub-formula. Nevertheless, after evaluating the sub-formula, the Partial/filt approach detects that there are no general winning strategies in the initial states and can directly conclude. All in all, the Partial/filt approach performs worse than Partial because it considers all reachable states at once, instead of different subsets of the reachable states separately, leading to many more strategies to check. In the case of 4 horses, the Partial/filt approach does not succeed in checking all the strategies for the sub-formula within 30 minutes, while Partial does.

Regarding the early approaches, the number of strategies to check in the whole set of reachable states (as done by the Early/filt approach) is larger than the strategies checked by the Early approach, since the Early strategies already made a choice in the initial states. This allows the Early approach to conclude for 4 horses within 30 minutes while Early/filt does not.

Regarding the Symbolic approaches, it is a coincidence that both approaches take a similar amount of time. The Symbolic approach has to evaluate the two strategic sub-formulas. On the other hand, Symbolic/filt has to perform pre-filtering on the sub-formula and then evaluate it, but does not have to evaluate the top formula.

Regarding all the approaches, the Partial approach takes a substantial amount of time because it has to check a large number of strategies to determine that none of them are winning. The Early approach has less work to do as it does not extend the strategies. Finally, the symbolic ones are better because they can evaluate all the strategies at once and determine that they are not winning.

Pre-filtering does not work for the partial and early approaches because, in both cases, it triggers the evaluation of the sub-formula on all reachable states, leading to many more strategies to consider, as these strategies have to take all states into account at the same time.

*4.1.3 ⟪Tian Ji⟫**X** Tian Ji null score*

Figure 6 shows the evolution of verification time of the six approaches for the formula ϕ_3^T on the model of Tian Ji. This formula encodes the fact that Tian Ji can enforce to lose the first race, and is false because even if he chooses his slowest horse, the king could use his slowest one, too.

Pre-filtering Pre-filtering removes a lot of moves. Indeed, there are only few states in which Tian Ji can keep his score at 0. In these states, he has to have a score of 0, and he has to still have a horse that surely loses the next race. Furthermore, pre-filtering for the Partial/filt and Early/filt approaches is even more efficient as they restrict the sub-formula *Tian Ji null score* to the successors of the initial states.

Partial approaches Thanks to the restriction to partial strategies, the Partial approach has fewer strategies to check before concluding that the formula is false.

Furthermore, thanks to the very efficient pre-filtering, the Partial/filt approach has very few strategies to check: there remains only $N-1$ strategies, as only the initial moves of the game are kept (except the one playing the best horse, as it cannot be used to win the first race). In this case, most of the time ($50-70\%$) is spent to perform pre-filtering.

Early approaches The Early approach has to check the N initial strategies to conclude that the formula is false. Indeed, it is not necessary to extend them as the approach can directly conclude that there is no winning strategy. Nevertheless, it has to complete each strategy with the compatible reachable moves to check that they are not winning, leading to extra work compared to the Partial/filt approach.

The Early/filt approach computes the same pre-filtering as the Partial/filt approach. It thus still has to check $N-1$ strategies before concluding that the formula is false. As for the Early approach, it has to complete each strategy with the compatible reachable moves to check that they are not winning.

Symbolic approaches The Symbolic approach encodes all strategies for Tian Ji, and checks them all at the same time, while Symbolic/filt benefits from pre-filtering and encodes fewer strategies.

Comparison The approaches with pre-filtering perform better than their counterpart without pre-filtering. This is due to the fact that the number of remaining strategies is significantly smaller than the initial number of strategies.

Furthermore, the Early/filt approach performs worse than Partial/filt because it has more work to do: for each initial move, it has to extend the strategy with compatible reachable moves before concluding that the strategy cannot win. On the other hand, the Partial/filt approach completes the strategy with pre-filtered moves only, and there are no such moves as pre-filtering is limited to the initial states.

The Partial approach performs worse than Early because it has to check partial strategies, while the Early approach only has to check the N initial moves (and complete them before checking them).

4.2 The three castles

This section presents the results for the properties of the three castles.

*4.2.1 ⟪Castle$_1$, Castle$_2$⟫**F** Castle$_3$ defeated*

Figure 7 shows the evolution of verification time of the six approaches for checking the formula ϕ_1^C on the model of the castles. The size of the model (Number of workers) is given as a triplet $\langle 1\ 2\ 3\rangle$, meaning that the first castle is defended by one worker, the second one by two and the third one by three workers. The tests were performed on instances in which there are at least as many workers in the first two castles as in the third castle. Cases in which there are more workers in the third castle than in the other two have not been considered because, in this case, the formula is false even with perfect information.

Pre-filtering Pre-filtering removes from 18% ($\langle 1\ 1\ 1\rangle$ case) to 77% ($\langle 1\ 1\ 2\rangle$ case) of the moves. For the other sizes, the gain of pre-filtering is between these two bounds.

The huge gain of 77% in the $\langle 1\ 1\ 2\rangle$ is explained by the fact that the power of the workers of the first two castles is

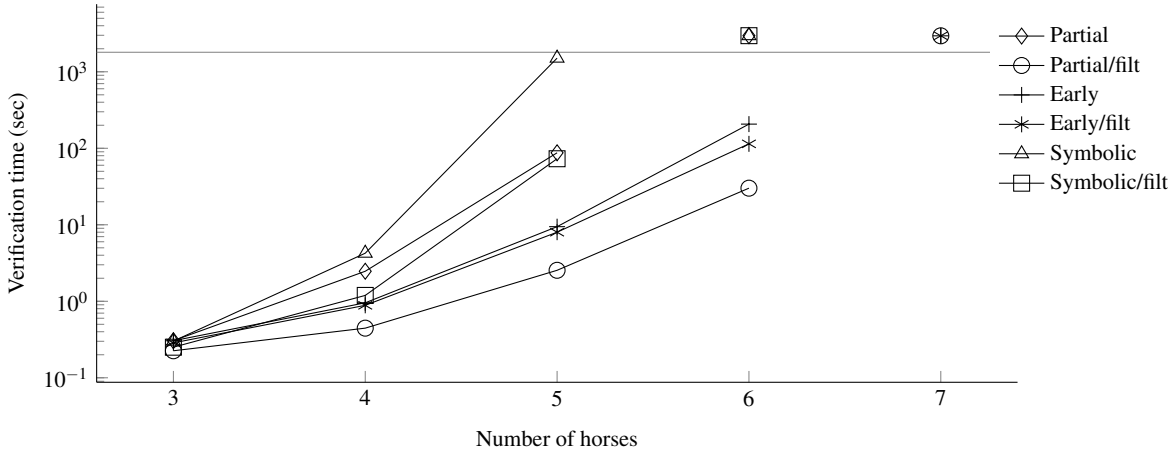

Fig. 6 Evolution of the verification time of the approaches for the formula $\phi_3^T = \langle\langle \mathit{Tian\ Ji} \rangle\rangle \mathbf{X}\ \mathit{Tian\ Ji\ null\ score}$.

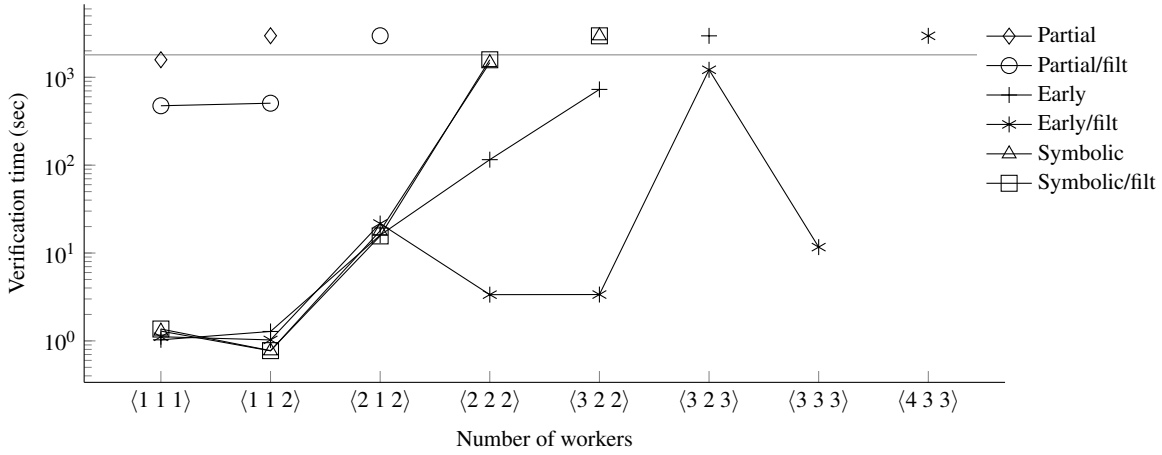

Fig. 7 Evolution of the verification time of the approaches for the formula $\phi_1^C = \langle\langle \mathit{Castle}_1, \mathit{Castle}_2 \rangle\rangle \mathbf{F}\ \mathit{Castle}_3\ \mathit{defeated}$.

comparable to the power of the workers of the third castle. This means that the first two castles workers do not have many winning moves, even if they know whether their opponents defended their castle before and the actual health points of the castles. On the other hand, in the ⟨1 1 1⟩ case, the first two castles have more power compared to the third one and can more easily win if they have perfect information. The other cases are between these two extremes. In all cases, the time needed to perform pre-filtering is negligible compared to the search for a winning strategy.

Partial approaches The Partial approach succeeds in finding a winning strategy within 30 minutes for the ⟨1 1 1⟩ case. Nevertheless, for the ⟨1 1 2⟩ case, it cannot find a winning one. In this case, there is the same number of strategies, but it is more costly to check each strategy as the model is bigger. On the other hand, the Partial/filt approach benefits from pre-filtering and finds a winning strategy more quickly than Partial. Nevertheless, it fails at finding a winning one in the ⟨2 1 2⟩ case.

Early In the ⟨1 1 1⟩ case, the Early approach needs to reach up to half the depth of the model to determine the strategies to be losing. This allows the approach to find a winning strategy easily.

In the ⟨1 1 2⟩ case, it needs to split only a few steps to determine that the strategies are losing. This is because the group of agents of the first two castles have relatively less power than in the previous case.

In the larger cases, the Early approach needs to reach about half way, again. The number of strategies increases with the number of workers to consider, as well as the time needed to check larger models.

Early/filt In the ⟨1 1 1⟩ and ⟨1 1 2⟩ cases, the Early/filt approach has fewer strategies to check than the Early approach because it benefits from pre-filtered moves.

In the ⟨2 1 2⟩ case, the approach finds a winning strategy within the same time as the Early approach.

In the ⟨2 2 2⟩, ⟨3 2 2⟩ and ⟨3 3 3⟩ cases, it very quickly finds a winning strategy (after resp. 18, 37 and 52 strategies). It benefits substantially from pre-filtering and finds a good

strategy after a few steps. It still has to reach about half way to find this winning strategy. In the $\langle 3\ 2\ 3 \rangle$ case, it needs to consider many more strategies before finding a good one.

The Early/filt approach benefits from pre-filtering and decreases the number of strategies to check, compared to the Early approach, but still struggles to find winning strategies for some cases such as the $\langle 3\ 2\ 3 \rangle$ and $\langle 4\ 3\ 3 \rangle$ cases.

Symbolic approaches The Symbolic approach has to encode and check all strategies at the same time. As the number of workers increases, there are more and more strategies for the group. Furthermore, the Symbolic/filt approach cannot benefit from pre-filtering because all equivalence classes are still present and all actions are still possible in each of them. Thus, it performs exactly as Symbolic, as the time to perform pre-filtering is negligible.

Comparison The partial approaches succeed in decreasing the number of uniform strategies to consider, and thanks to early termination, can stop as soon as a winning strategy is found. Nevertheless, the number of partial strategies to consider is still large, and the approaches quickly fail to find a winning one.

The symbolic approaches are better. Nevertheless, pre-filtering does not benefit to the Symbolic/filt approach, thus both approaches do the same work.

The early approaches are the best in the present scenario because they can quickly determine that a partial strategy and all its extensions cannot be winning. In particular, the Early/filt approach benefits from pre-filtering and drastically reduces the number of strategies it checks for the largest models. The Early/filt approach shows some irregularities in performances because it sometimes makes the right choices of actions, and sometimes not.

4.2.2 $\langle\langle Worker_1, Worker_2 \rangle\rangle \boldsymbol{F}$ *all defeated*

Figure 8 shows the evolution of verification time of the six approaches for checking the formula ϕ_2^C on the model of the castles. This formula is false for all checked sizes.

Pre-filtering A major difference between the $\langle 1\ 1\ 1 \rangle$ case and the others is that, in the former case, the two workers have a strategy to achieve their goal when they have perfect information, while it is not the case for the greater sizes. Thus, pre-filtering, in the cases of larger models, allows the Partial/filt and Early/filt approaches to directly determine that the formula is false, without checking any strategy.

Partial approaches The Partial approach reaches the time-out even for the smallest model size. Given the number of possible strategies (6.9×10^9) and the fact that the approach must check them all to determine that the formula is false, this result is not surprising.

On the $\langle 1\ 1\ 1 \rangle$ case, pre-filtering drastically reduces the number of moves to consider, and thus the number of strategies the Partial/filt approach needs to check before stating that the formula is false. The approach shows that the remaining strategies are losing within the time limit, while Partial fails to do so. For the other cases, pre-filtering performs all the work.

Early For the $\langle 1\ 1\ 1 \rangle$ case, the Early approach needs to reach about half way from the initial state to determine strategies to be losing, as for the previous formula. This allows the approach to check all strategies more easily.

For the other cases, the approach only needs to check the 16 initial actions of the two workers to conclude that there can be no winning strategy. Indeed there are no winning strategy in these cases, even with perfect information, and the approach can determine it directly.

The increasing of time for the Early approach only comes from the fact that the model is bigger and bigger, making the verification of these 16 strategies longer and longer.

Early/filt For the $\langle 1\ 1\ 1 \rangle$ case, the Early/filt approach does not gain from pre-filtering. In fact, the moves that are filtered out are never reached by the Early/filt approach because it can determine that the strategies are losing before reaching them. Thus, it behaves like Early on this case. For the other cases, the Early/filt approach does not check any strategy since pre-filtering directly determines that there can be no winning strategy.

Symbolic approaches The Symbolic approach behaves in the same way for all model sizes. The only differences come from building a model of increasing size. The actual fixpoint computation to determine the winning strategies is the same in all cases. On the other hand, the Symbolic/filt approach gains from pre-filtering. It drastically reduces the number of strategies to encode for the first two cases. For the three last ones, there remains only one strategy to encode and check.

Comparison The Partial approach does not handle the smallest model because it has to check the huge number of strategies to determine that there are no winning ones. On the other hand, the Partial/filt and Early/filt approaches only need pre-filtering to conclude. The Early approach can also quickly determine that the formula is false because it just needs to check all possible actions in the initial state. The symbolic approaches also perform well because the BDDs they compute remain very small.

In conclusion, all approaches are comparable for the case $\langle 2\ 1\ 2 \rangle$ and after because it is easy to show that the formula is false, except for the Partial approach that must check all possible strategies to reach this conclusion.

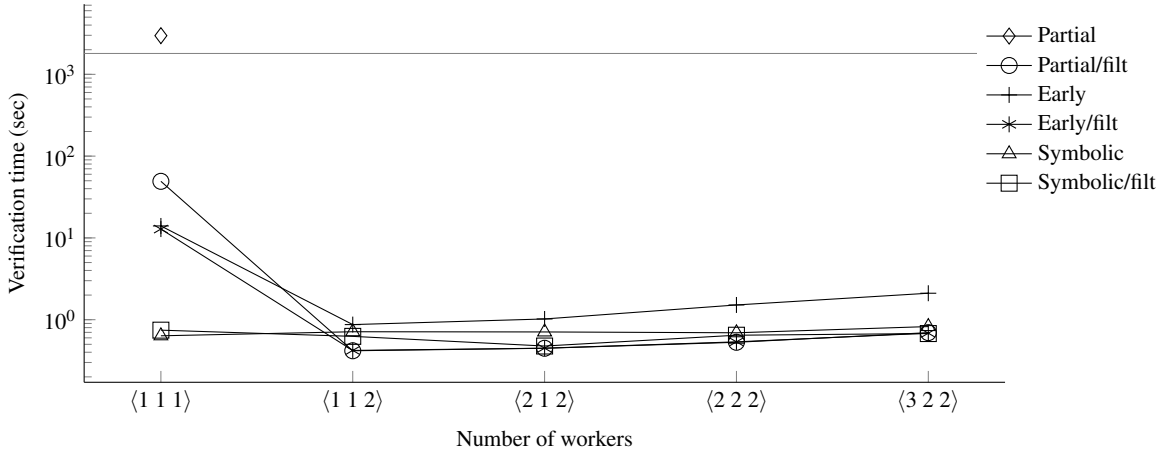

Fig. 8 Evolution of the verification time of the approaches for the formula $\phi_2^C = \langle\!\langle Worker_1, Worker_2 \rangle\!\rangle \mathbf{F}$ *all defeated*.

4.3 The prisoners and the light bulb

This section presents the results for the property

$\langle\!\langle prisoners \rangle\!\rangle [\neg executed \ \mathbf{U} \ released]$

of the prisoners model. Figure 9 shows the evolution of verification time of the six approaches for checking the formula on the model of the prisoners. The formula is true in all tested models, showing that the prisoners effectively have a strategy to be released without being executed. The number of partial strategies from the initial state grows exponentially in terms of the number of prisoners. Furthermore, the number of strategies is already huge for the smallest model, compared to the model of Tian Ji: the 2 prisoners have ≈ 20000 strategies while Tian Ji has only 24 strategies with 3 horses.

Pre-filtering Pre-filtering does not remove a lot of moves. Indeed, when the prisoners have perfect information, the counting one knows who has already been interrogated and can make a correct announcement as soon as possible. The only moves that are removed are those leading to an incorrect announcement and the execution of all prisoners.

Partial approaches The Partial approach is lucky to find a winning strategy for 2 prisoners. For 3 prisoners, it does not find a good one within 30 minutes. Partial/filt benefits from pre-filtering and even achieves to find a winning strategy for 3 prisoners, but fails for 4 prisoners.

Early approaches The Early approach is quicker than the Partial approach to find a winning strategy for 2 prisoners. It is explained by the fact that as soon as the current strategy considers an incorrect announcement, the approach stops extending it because it is surely losing. Early/filt performs even better for 2 prisoners than the Early approach, but it is the opposite for 3 prisoners. As the time needed to perform pre-filtering is negligible, this simply means that the Early approach makes better decisions than Early/filt for 3 prisoners.

Symbolic approaches The Symbolic approach is very efficient for 2 prisoners, but is less for 3. This is because there are sufficiently few strategies to encode in the former case, but too many for the latter. The other approaches that succeed in finding winning strategies for 3 prisoners simply made the right choices, as they are not able to check all possible strategies within 30 minutes, while the symbolic approaches have to check them all. The Symbolic/filt approach behaves like Symbolic because pre-filtering does not remove any equivalence classes, nor any actions in these classes.

Comparison As for the previous model, the number of strategies to consider is huge. The Partial approach fails at finding a winning strategy for 3 prisoners.

The number of partial strategies to check by the Partial/filt approach is lower than for Partial. This means that pre-filtering is useful here. On the other hand, the partial approaches are less efficient than the early ones because the latter can rule out strategies more easily. Both early approaches behave similarly.

Finally, the symbolic approaches both behave in the same way, and are very efficient when considering the strategies for 2 prisoners. Nevertheless, for 3 prisoners, the number of strategies is too large, and the other approaches make the right choices and find a winning strategy within the time limit, while the symbolic approaches have to consider all strategies at once to conclude.

4.4 BDD variable reordering techniques

The implementation of the approaches is based on BDDs. It is well-known that ordered binary decision diagrams performances are highly correlated to the order of their variables.

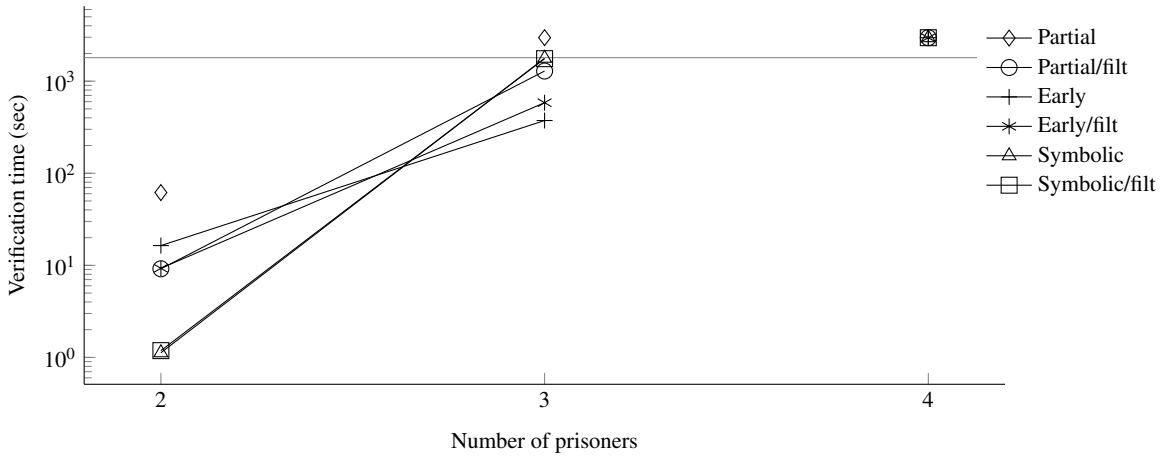

Fig. 9 Evolution of the verification time of the approaches for the formula $\phi^P = \langle\langle prisoners \rangle\rangle[\neg executed \, \mathbf{U} \, released]$.

The implementation being based on NuSMV, it inherits all the functionalities of the tool to dynamically reorder the BDD variables. NuSMV proposes 19 different heuristics for dynamic reordering. To assess their performances, 18 of them have been tested with the approaches (see the NuSMV Manual for more information about the different reordering heuristics [11]). The omitted one is the *exact* heuristic that computes the optimal order. It can take a lot of time and is not advised with more than 16 Boolean variables [11]. The 18 heuristics have been tested on different formulas, one for each approach. These formulas have been chosen such that the time needed to solve the model-checking problem is not too low—when it is too low, other quick computations such as building the model can have a significant impact on the overall model-checking time—, and not too high—hitting the 1800 second timeout would yield no useful information for comparing the heuristics. As for the previous tests, each formula has been checked 20 times. The formulas are

- ϕ_1^T with 6 horses for both *Partial* and *Partial/filt* approaches;
- ϕ_2^T with 4 horses for the *Early* approach;
- ϕ_3^T with 4 horses for the *Early/filt* approach;
- ϕ_1^C with 2, 1 and 2 workers for the *Symbolic* and *Symbolic/filt* approaches.

Figure 10 shows the time needed for the Early approach to verify the formula

$$\langle\langle Tian\ Ji \rangle\rangle \mathbf{F} \, \langle\langle Tian\ Ji \rangle\rangle[\neg King\ wins \, \mathbf{U} \, Tian\ Ji\ wins]$$

on the model of Tian Ji with 4 horses.

First, these tests show that no heuristic is better than another. The approach takes about 120 seconds to verify the formula, regardless of the reordering technique. Second, these tests show the variability stays small: the approach usually needs from 115 to 130 seconds to perform the verification.

The other tested approaches showed similar results. For all reordering heuristics, the partial, *Early/filt* and symbolic approaches took about 100 seconds. No heuristic showed better performances in verifying the formulas, with all tested model sizes and approaches. This leads to the conclusion that the chosen heuristic is not important in the present cases and that choosing the default *sift* one is a sensible choice. This heuristic has been used for all the other tests presented in this paper.

4.5 Conclusions on the experiments

Based on the observations made on the experiments presented above, we can draw some general conclusions.

The best approach to check that there exists a winning strategy when most of them are winning is the Partial approach. Nevertheless, it performs poorly to show that there are no or few winning strategies.

The early approaches present a better trade-off since they take more time to show that there is a winning strategy if most of them are winning, but can more easily find one when there are only few winning ones, or even show that there are no winning strategies. Nevertheless, in the case in which only complete partial strategies are winning, and not some incomplete ones, the early approaches tend to perform extra work that is not needed. Indeed, they have to extend a strategy completely to find a winning one, and the intermediate computations of losing and winning states do not yield any gain.

The symbolic approaches work better when there is a huge number of strategies to consider because they can represent them in a compact way. On the other hand, the other approaches cannot handle a huge amount of strategies since they need to enumerate them. Furthermore, the symbolic approaches work well with nested strategic formulas.

Pre-filtering may or may not help. Either it removes a large number of losing moves when there are big parts of the model in which the agents have no winning strategies at all

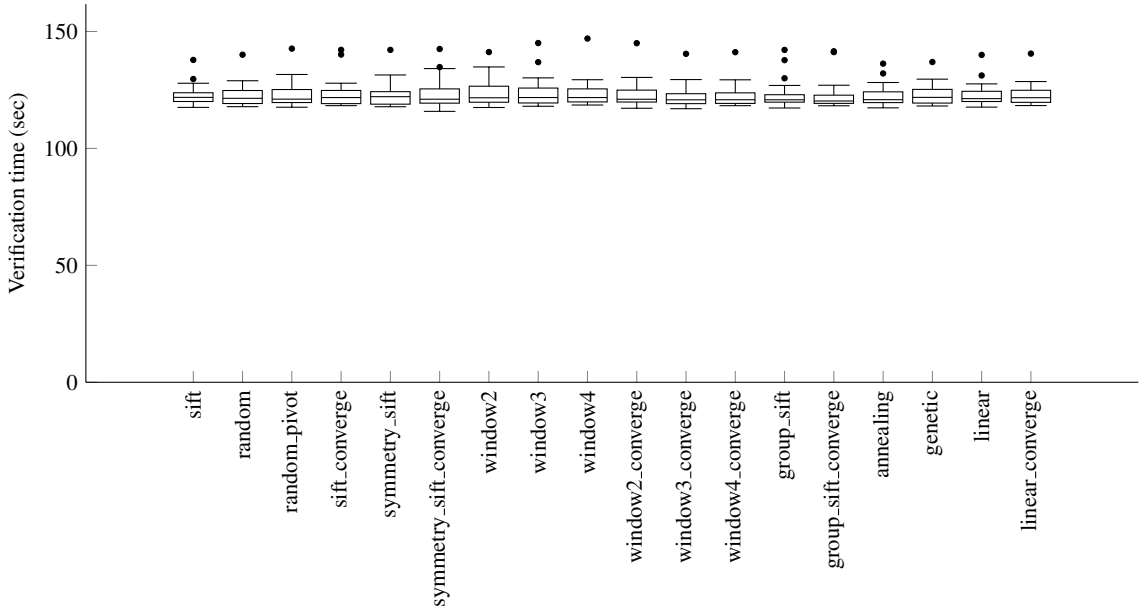

Fig. 10 Time taken with all variable reordering heuristics with the *Early* approach for checking formula ϕ_2^T on the model of Tian Ji with 4 horses.

(even non-uniform ones), or it removes only a few of them, producing extra work that is not needed. In some cases, it can even directly conclude that there are no winning strategies, avoiding the need to check any strategy.

The symbolic approaches are more stable than the others, in the sense that the time needed to perform the model checking is less dependent on the property being checked. This can be seen on the first two formulas of the model of Tian Ji: the Symbolic approach can evaluate them for 4 horses but never for 5 in the available time. On the third formula, the $\langle\langle \Gamma \rangle\rangle \mathbf{X}$ operator is easier to deal with because the fixpoint computation is simpler, allowing the approach to evaluate the formula for 5 horses. On the other hand, the other approaches have very variable performances depending on the formula.

While the approaches can be efficient in finding winning strategies when there are a lot of them, or to check that there are no winning strategies when there are not even non-uniform ones, they can be very unpredictable when there are only few winning strategies. Partial and early approaches can take a long time to find the winning combinations of moves, and symbolic approaches have to deal with more complicated BDDs.

While NuSMV and PyNuSMV propose some heuristics to dynamically reorder the BDD variables, the tests showed that none of them is better than the others.

In summary, there is no best approach for all cases. The easiest way to solve a model-checking problem is thus to run all approaches in parallel. We can also build some guidelines from the conclusions above to try to identify the best approach depending on different criteria:

- If most of the uniform strategies of the agents should be winning, the partial approaches are the best to find one.
- If it is not clear whether there are many winning strategies, then the early approaches present a better trade-off. They are able to find winning strategies, but also show that there are none.
- The symbolic approaches are a better choice if the number of strategies is really huge, or when the property contains nested strategic formulas.
- If many moves of the game do not belong to winning general strategies, then pre-filtering should reduce the number of strategies to check.
- But if the agents can easily win the game when observing the whole state (for instance, if they can observe the solution of a puzzle, or the code for a safe), then pre-filtering is not useful.

Most of these guidelines rely on features of the model that an automated tool cannot easily grasp. The responsibility is thus on the user to choose the approach to use. Furthermore, these guidelines are based on the experiments presented in this paper, and these experiments showed that the approaches can be unpredictable when there are only few winning strategies.

5 Conclusion

Starting from *Alternating-time Temporal Logic* (*ATL*), reasoning about strategies of the agents of a system has been investigated for a long time. In particular, some logics focus on reasoning about the strategies that the agents can actually play based on their current observations of the system. While these logics have been studied for 15 years, practi-

cal approaches to solve their model-checking problem only appeared recently.

This paper briefly described the three approaches of Pilecki et al. [27], Huang and van der Meyden [20,19], and Busard et al. [8], adapted to $ATLK_{irF}$, an extension of ATL with memoryless uniform strategies and unconditional fairness constraints on states [9]. It also presented their implementation with PyNuSMV, a Python library for prototyping BDD-based model-checking algorithms based on NuSMV [6]. Their implementation in the same BDD-based framework, within the same tool, lead to a fair comparison of the techniques on practical examples. More precisely, this paper described three models with strategic formulas used to compare the practical performances of the different approaches. These experiments showed that the three approaches have their own advantages and disadvantages, and none of them effectively surpassed the others on all the experiments. From these experiments, some guidelines have been proposed to choose the approach when facing a new model-checking problem.

The experiments also showed that none of the approaches is scalable: none of them succeeded for Tian Ji's problem with more than 6 horses, for the castles problem with more than 9 workers, and for the prisoners' problem with more than 3 prisoners. The problem of model checking strategies under imperfect information, already known to be theoretically complex[1], seems to be complex in practice, too.

5.1 Related work

Several other approaches have been proposed to deal with problems similar to the ones of this paper.

Calta et al. Calta et al. propose an algorithm to check the existence of uniform strategies over sets of states of iCGS, that is, iCGSf without fairness constraints [10]. Nevertheless, it is difficult to adapt their approach to the case of $ATLK_{irF}$, to take fairness constraints into consideration, and to implement it in a BDD-based framework.

Lomuscio and Raimondi Lomuscio and Raimondi propose to perform the model checking of uniform strategies by enumerating all Γ-uniform systems compatible with a given interpreted system. Such a Γ-uniform system compatible with the original model is a restriction of the system where agents of Γ choose only one action in each equivalence class. They then conclude that the original system satisfies the formula if there exists a Γ-uniform compatible system satisfying the formula [26].

[1] The problem of model checking strategies under imperfect information is Δ_2^P-complete [22,23].

Nevertheless, the semantics they handle is different from $ATLK_{irF}$. In the case of Lomuscio and Raimondi, the strategies must be winning for the whole formula, that is, the same uniform strategy must be winning for all strategic subformulas, and for all states of interest. On the other hand, $ATLK_{irF}$ semantics is local to the sub-formula, and two different formulas or distinguishable states can have a different winning strategy. $ATLK_{irF}$ semantics follows the ideas of ATL, where two strategic sub-formulas can be satisfied because of two different strategies, while the semantics of Lomuscio and Raimondi diverges from it. Furthermore, their idea is similar to the first idea proposed by Busard et al. [7, 9], shown to be highly ineffective compared to the partial approach [8].

Observation-based two-player games Raskin et al. propose an approach to check the existence of winning observation-based strategies with perfect recall in the context of two-player games [28]. Their algorithm is based on fixpoint computations on sets of states and works with anti-chains. The objectives of their strategies are ω-regular ones, such as Büchi and co-Büchi objectives.

Similarly, Bozianu et al. propose another anti-chain based algorithm for checking the existence and synthesising strategies with imperfect information and perfect recall in the context of two-player games [2]. The objectives of their strategies are defined using an extension of LTL with knowledge operators.

These problems are related to the problem of $ATLK_{irF}$ model checking. Nevertheless, they put different limitations on the problem. $ATLK_{irF}$ is limited to memoryless strategies. The approaches of Raskin et al. and Bozianu et al. limit the games to be two-player games. In this context, checking the existence of winning memory-full uniform strategies is decidable. Furthermore, the approaches they propose, based on anti-chains, are far from the BDD-based algorithms used in this paper.

5.2 Future work

There remain many ways to go further than the experiments of this paper.

Experiments with real-life cases The tested models and formulas are toy models based on the ones found in the literature. They do not necessarily reflect the structure, size and complexity of real-life cases. It would be interesting to find real-life cases of iCGSf and to test the approaches on them.

Investigating the initial order of BDD variables Section 4.4 showed that, while NuSMV and PyNuSMV propose several heuristics to dynamically reorder the BDD variables during

the process, none of these heuristics had a significant impact on the performances. Nevertheless, there exist other ways to improve the performance of BDD-based algorithms by manipulating this variable order. In particular, the initial order of the variables plays a big role in the successive orders computed by the heuristics. For the experiments of this paper, the standard initial order defined by the order in which the variables have been declared in the description of the model has been used. It would be interesting to evaluate the impact of this initial order and to define, if possible, general guidelines for specifying efficient initial orders.

Mixing approaches This paper showed that the partial, early and symbolic approaches are comparable in efficiency, and that they are efficient in different situations. It would be interesting to mix them, to evaluate different sub-formulas using different approaches. For instance, given the formula $\langle\langle \Gamma_1 \rangle\rangle \mathbf{F}\ p \wedge \langle\langle \Gamma_2 \rangle\rangle \mathbf{G}\ q$, the first sub-formula $\langle\langle \Gamma_1 \rangle\rangle \mathbf{F}\ p$ could be evaluated with the partial approach while the second one $\langle\langle \Gamma_2 \rangle\rangle \mathbf{G}\ q$ could be evaluated with the early approach. The approach to use for each sub-formula could be defined by the user, but it would also be interesting to investigate criteria to automatically select the best approach.

References

1. Alur, R., Henzinger, T.A., Kupferman, O.: Alternating-time temporal logic. J. ACM **49**(5), 672–713 (2002). DOI 10.1145/585265.585270
2. Bozianu, R., Dima, C., Filiot, E.: Safraless synthesis for epistemic temporal specifications. In: A. Biere, R. Bloem (eds.) Computer Aided Verification, *Lecture Notes in Computer Science*, vol. 8559, pp. 441–456. Springer International Publishing (2014). DOI 10.1007/978-3-319-08867-9_29
3. Bryant, R.E.: Graph-based algorithms for boolean function manipulation. IEEE Transactions on Computers **100**(8), 677–691 (1986)
4. Bulling, N., Jamroga, W.: Comparing variants of strategic ability: how uncertainty and memory influence general properties of games. Autonomous Agents and Multi-Agent Systems **28**(3), 474–518 (2014). DOI 10.1007/s10458-013-9231-3
5. Busard, S.: Symbolic model checking of multi-modal logics: uniform strategies and rich explanations. Ph.D. thesis, Université catholique de Louvain (2017)
6. Busard, S., Pecheur, C.: PyNuSMV: NuSMV as a Python library. In: G. Brat, N. Rungta, A. Venet (eds.) Nasa Formal Methods 2013, *LNCS*, vol. 7871, pp. 453–458. Springer-Verlag (2013)
7. Busard, S., Pecheur, C., Qu, H., Raimondi, F.: Reasoning about strategies under partial observability and fairness constraints. In: F. Mogavero, A. Murano, M.Y. Vardi (eds.) Proceedings 1st International Workshop on Strategic Reasoning, SR 2013, Rome, Italy, March 16-17, 2013., *EPTCS*, vol. 112, pp. 71–79 (2013). DOI 10.4204/EPTCS.112.12. URL `http://dx.doi.org/10.4204/EPTCS.112.12`
8. Busard, S., Pecheur, C., Qu, H., Raimondi, F.: Improving the model checking of strategies under partial observability and fairness constraints. In: S. Merz, J. Pang (eds.) Formal Methods and Software Engineering, *Lecture Notes in Computer Science*, vol. 8829, pp. 27–42. Springer International Publishing (2014). DOI 10.1007/978-3-319-11737-9_3
9. Busard, S., Pecheur, C., Qu, H., Raimondi, F.: Reasoning about memoryless strategies under partial observability and unconditional fairness constraints. Information and Computation **242**, 128 – 156 (2015). DOI 10.1016/j.ic.2015.03.014
10. Calta, J., Shkatov, D., Schlingloff, H.: Finding uniform strategies for multi-agent systems. In: J. Dix, J. Leite, G. Governatori, W. Jamroga (eds.) Computational Logic in Multi-Agent Systems, *Lecture Notes in Computer Science*, vol. 6245, pp. 135–152. Springer Berlin / Heidelberg (2010). DOI 10.1007/978-3-642-14977-1_12
11. Cavada, R., Cimatti, A., Jochim, C.A., Keighren, G., Olivetti, E., Pistore, M., Roveri, M., Tchaltsev, A.: NuSMV 2.5 user manual
12. Cimatti, A., Clarke, E., Giunchiglia, E., Giunchiglia, F., Pistore, M., Roveri, M., Sebastiani, R., Tacchella, A.: NuSMV 2: An open-source tool for symbolic model checking. In: E. Brinksma, K.G. Larsen (eds.) Computer Aided Verification, *Lecture Notes in Computer Science*, vol. 2404, pp. 359–364. Springer Berlin Heidelberg (2002). DOI 10.1007/3-540-45657-0_29
13. Clarke, E.M., Grumberg, O., Peled, D.: Model Checking. MIT Press (1999)
14. Dastani, M., Jamroga, W.: Reasoning about strategies of multi-agent programs. In: Proceedings of AAMAS 10, pp. 997–1004 (2010)
15. Dima, C., Tiplea, F.L.: Model-checking ATL under imperfect information and perfect recall semantics is undecidable. CoRR **abs/1102.4225** (2011)
16. van Ditmarsch, H., Knight, S.: Partial information and uniform strategies. In: N. Bulling, L. van der Torre, S. Villata, W. Jamroga, W. Vasconcelos (eds.) Computational Logic in Multi-Agent Systems, *Lecture Notes in Computer Science*, vol. 8624, pp. 183–198. Springer International Publishing (2014). DOI 10.1007/978-3-319-09764-0_12
17. van Ditmarsch, H., Kooi, B.: One hundred prisoners and a light bulb. In: One Hundred Prisoners and a Light Bulb, pp. 83–94. Springer International Publishing (2015). DOI 10.1007/978-3-319-16694-0_9
18. Fagin, R., Halpern, J.Y., Moses, Y., Vardi, M.Y.: Reasoning about Knowledge. MIT Press, Cambridge (1995)
19. Huang, X., van der Meyden, R.: An epistemic strategy logic (extended abstract). In: F. Mogavero, A. Murano, M.Y. Vardi (eds.) Proceedings 2nd International Workshop on Strategic Reasoning, Grenoble, France, April 5-6, 2014, *Electronic Proceedings in Theoretical Computer Science*, vol. 146, pp. 35–41. Open Publishing Association (2014). DOI 10.4204/EPTCS.146.5
20. Huang, X., van der Meyden, R.: Symbolic model checking epistemic strategy logic. In: Proceedings of the Twenty-Eighth AAAI Conference on Artificial Intelligence, July 27 -31, 2014, Québec City, Québec, Canada., pp. 1426–1432 (2014)
21. Jamroga, W., Ågotnes, T.: Constructive knowledge: what agents can achieve under imperfect information. Journal of Applied Non-Classical Logics **17**(4), 423–475 (2007). DOI 10.3166/jancl.17.423-475
22. Jamroga, W., Dix, J.: Model checking abilities under incomplete information is indeed Δ_2^P-complete. In: EUMAS'06 (2006)
23. Jamroga, W., Dix, J.: Model checking abilities of agents: A closer look. Theory of Computing Systems **42**(3), 366–410 (2008). DOI 10.1007/s00224-007-9080-z
24. Jamroga, W., van der Hoek, W.: Agents that know how to play. Fundamenta Informaticae **Volume 63**(2), 185–219 (2004)
25. Lomuscio, A., Qu, H., Raimondi, F.: MCMAS: an open-source model checker for the verification of multi-agent systems. International Journal on Software Tools for Technology Transfer pp. 1–22 (2015). DOI 10.1007/s10009-015-0378-x
26. Lomuscio, A., Raimondi, F.: Model checking knowledge, strategies, and games in multi-agent systems. In: 5th International Joint Conference on Autonomous Agents and Multiagent Systems (AAMAS 2006), Hakodate, Japan, May 8-12, 2006, pp. 161–168 (2006). DOI 10.1145/1160633.1160660

27. Pilecki, J., Bednarczyk, M.A., Jamroga, W.: Synthesis and verification of uniform strategies for multi-agent systems. In: N. Bulling, L. van der Torre, S. Villata, W. Jamroga, W. Vasconcelos (eds.) Computational Logic in Multi-Agent Systems, *Lecture Notes in Computer Science*, vol. 8624, pp. 166–182. Springer International Publishing (2014). DOI 10.1007/978-3-319-09764-0_11
28. Raskin, J., Chatterjee, K., Doyen, L., Henzinger, T.A.: Algorithms for omega-regular games with imperfect information. Logical Methods in Computer Science **3**(3) (2007). DOI 10.2168/LMCS-3(3:4)2007
29. Schobbens, P.Y.: Alternating-time logic with imperfect recall. Electronic Notes in Theoretical Computer Science **85**(2), 82 – 93 (2004). DOI 10.1016/S1571-0661(05)82604-0